SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

PERSONOITAVA YHTEENVETONÄKYMÄ GRAAFISISTA RAPOR-TEISTA

TEKIJÄ: Jasmin Hiltunen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä Jasmin Hiltunen	
Työn nimi Personoitava yhteenvetonäkymä graafisista raporteista	
Päiväys 12.5.2015	Sivumäärä / Liitteet 33 / 2
Ohjaajat Lehtori Jussi Koistinen, lehtori Sami Lahti	
Toimeksiantaja / Yhteistyökumppani Enfo Oyj, Kuopio	

Tiivistelmä

Tämän opinnäytetyön aiheena oli toteuttaa Enfo Oyj:lle palvelupäälliköiden ja Service Deskin esimiesten käyttöön näkymä, joka toimii osana Service Deskin omaa informaatio- ja raporttiportaalia. Portaalin esittämää graafista raporttitietoa haluttiin saada esille yhteen näkymään, jotta tiedot olisivat helposti nähtävissä ja vertailtavissa yhdellä silmäyksellä ilman raportista toiseen navigointia. Graafinäkymän tuli olla helposti personoitavissa, jotta kukin käyttäjä saa katsottavaksi haluamansa raportit. Sen vuoksi portaaliin toteutettiin graafinäkymän lisäksi myös käyttäjän autentikointi ja käyttäjäroolit.

Työ oli web-pohjainen ja se toteutettiin osaksi jo olemassa olevaa portaalia Microsoftin Visual Studiota käyttäen. Sovelluskehyksenä oli ASP.NET MVC 4 ja kehityskielenä C#. Työssä punnittiin erilaisia tiedontallennus- ja autentikointimenetelmiä ja huomio kiinnitettiin graafinäkymän käytettävyyteen ja selkeyteen. Sen vuoksi työssä käytettiin HTML:n rinnalla runsaasti JavaScriptiä ja jQuerya toiminnallisuuksien luomiseen sekä Google Charts -ilmaistyökalua graafisen raporttitiedon esittämiseen. Lopputuotetta testattiin kolmella nykyisin yleisimmin käytössä olevalla internetselaimella.

Lopputuotteen ulkonäkö ja käytettävyys vastasivat hyvin odotuksia ja työn tavoitteet täyttyivät.

Avainsanat
ASP.NET MVC, C#, HTML, JavaScript, jQuery, Google Chart, raportti, graafinen

Julkinen

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author Jasmin Hiltunen			
Title of Thesis Individualizable Summary View of Graphical Reports			
Date	12 May 2015	Pages / Appendices	33 / 2
Supervisors Mr. Jussi Koistinen, Lecturer, Mr. Sami Lahti, Lecturer			
Client Organisation / Partners Enfo Ltd. Kuopio Finland			

Abstract

The purpose of this thesis was to create a simple application view for the Enfo Service Desk unit. The application view works as a part of the current web-based information & report portal that shows graphic reports of specific history data. The view gathers several individual reports together to be shown at one glance and it can be used in most web browsers. The view can be individualized to show only those reports that the user needs. Consequently, a way of user authentication and user roles were created besides the graphical view.

The application view is web-based and it was implemented using the ASP.NET MVC application framework including C#, HTML, JavaScript and jQuery languages. The Google Charts tool was used to show history data in an easy graphic way. The only needed development tool was Microsoft Visual Studio which was used for simple database management as well. Different options for data storing and user authentication were considered in this thesis. The main priority was in the simple usage and layout. The final product was tested using three of the most popular web browsers.

As a result, the main goal of this project was achieved and the final product meets the set layout and usability requirements.

Keywords
ASP.NET MVC, C#, HTML, JavaScript, jQuery, Google Chart, report, graphic

Public

ESIPUHE

Ohjelmistotekniikan amk-opintoni alkoivat jo vuonna 2005. Matka on siis ollut pitkä ja monivaiheinen, mutta nyt se on vihdoin saatu päätökseen.

Kiitän erityisesti Savonia-ammattikorkeakoulun lehtori Sami Lahtea loistavasta opoudesta, lehtori Jussi Koistista tämän opinnäytetyön ohjaamisesta sekä lehtori Veijo Pitkästä, joka kaksi vuotta sitten ylipäätään mahdollisti jatko-opintojeni sujuvuuden. Suuret kiitokset myös Kuopion Enfo Oyj:n Service Deskiin johtaja Kyösti Vähäkainulle ja spesialisti Sauli Ryynäselle tämän opinnäytetyön aiheesta.

Kuopiossa 12.5.2015
Jasmin Hiltunen

SISÄLTÖ

LYHENTEET JA MÄÄRITELMÄT

AD	Active Directory, Windows-palvelimen käyttäjätietokanta
Asetustaulu	Tietokannan taulu, joka sisältää graafinäkymän komponenttien tiedot, ”asetukset”
Autentikointi	Käyttäjän tunnistaminen, tähän työhön valittiin käyttäjätunnus ja salasana
Authorisointi	Käyttäjän pääsyn rajaaminen vain esim. tietyille sivuille tai sivualueille
Code First	Eräs ASP.NET MVC -kehitystapa, jonka avulla projektiin luodaan mallin mukainen uusi tietokanta, tai päivitetään jo olemassa oleva tietokanta vastaamaan päivitettyä mallia
CSS	Cascading Style Sheet, www-sivun ulkoasutiedosto
Google Chart	Googlen ylläpitämä ilmaistyökalu raporttitietojen graafiseen esittämiseen
Graafi	Työssä käytetty nimitys Google Chartille
Heuristinen arviointi	Kokemukseen perustuva arviointi, jolla on tarkoitus löytää käyttöliittymän toimivuuden ja käytettävyyden ongelmakohdat
Heksadesimaalijärjestelmä	Tietotekniikassa käytettävä kantalukujärjestelmä, jonka kantaluku on 16
HTML	Hyper-Text Markup Language, hypertekstin merkintäkieli, joka toimii mm. www-sivujen ohjelmointikielenä
JSON	JavaScript Object Notation, eräs tiedostonvälitystapa, joka mahdollistaa tiedonvälityksen lähes minkä tahansa ohjelmointikielten välillä
Komponentti	Työn graafinäkymän div-osio, joita on sivulla useampi
Lazy Loading	Minkä tahansa tietojen taustalataus, tässä työssä se koskee MVC-mallin viitattujen taulujen sisällön taustalatausta
Migraatio	Code First -kehitystavan mukainen päivitys, jolla luodaan uusi tietokanta tai päivitetään jo olemassa oleva tietokanta muutettua mallia vastaavaksi
MVC	Model-View-Controller -arkkitehtuuri

Osanäkymä	Partial view, näkymän osa
Portaali	Toimeksiantajan oma informaatio- ja raportointikanava, pohjaprojekti, johon opinnäytetyö tehtiin
Relaatio	Esimerkiksi tietokannan eri taulujen välinen yhteys
SVG	Scalable Vector Graphic, skaalautuva vektorigrafiikka, selainten käyttämä ja W3C:n ylläpitämä tekijänoikeusmaksuton grafiikkaformaatti
URL	Uniform Resource Locator, linkki, www-sivujen osoitekäytäntö
Widget	Suom. vempain tai vekotin, ohjelmistotekniikassa sovellus

1 JOHDANTO

Tämä opinnäytetyö on tehty Enfo Oyj:n Service Deskille Kuopioon. Työn tekeminen aloitettiin helmikuussa 2015 ja se valmistui noin kolme kuukautta myöhemmin.

Enfo Oyj:n Service Deskillä on käytössään sisäinen informaatio- ja raportointikanava, portaali, joka välittää katsojilleen ajantasais- ja historiatietoa työmääristä ja niiden kehityksestä helposti tulkittavina graafisina kuvaajina. Portaali on toteutettu tietotekniikan opinnäytetyönä vuosina 2013 - 2014 ja se on otettu Service Deskissä päivittäiseen käyttöön selainversion lisäksi myös seinänäyttönä. Portaalia on edelleen laajennettu ja kehitetty muun muassa ohjelmistotekniikan opiskelijoiden työharjoitteluina ja nyt myös opinnäytetyönä.

Tämän opinnäytetyön aihe pohjautuu Enfo Oyj:n Service Deskin esimiesten ideoihin ja tarpeisiin. Graafinäkymän tuleva pääasiallinen käyttäjäryhmä (yli 10 henkilöä) koostuu Service Deskin esimiehistä ja Enfon palvelupäälliköistä, jotka toimivat yhteistyössä asiakkaan tietohallinnon ja Enfon palvelutuotannon kanssa. He käyttävät portaalin tuottamia graafisia kuvaajia raportinkaltaisesti osana omaa työtään tarkastellessaan esimerkiksi tiimien puhelumäärien historiatietoja. Graafeja on tähän asti tarkasteltu selaimella yksitellen, joten jonkinlainen yhteisnäkymän luominen tuli tarpeeseen. Raporttien yhteisnäkymän eli graafinäkymän sisällön toivottiin olevan muokattavissa, koska jokaisella esimiehellä ja palvelupäälliköllä on omat seurattavat raporttinsa.

2 ENFO OYJ

Vuonna 1964 Kuopioon perustettiin Tietosavo Oy, jonka taustalla vaikutti seitsemän kuopiolaista kasvuyritystä. Tietosavo Oy järjesti 1960-luvulla tietotekniikan opetusta itse, koska osaavaa työvoimaa oli tarjolla vähemmän kuin vastaavaa työtä. Tietotekniikan kehittyminen 1970-luvulla vei yritystä kohti IT-alan huippua tuhatkertaistaen yrityksen osakepääoman. 1980-luvulla yritys kohdisti strategiansa järjestelmiin ja ohjelmistoihin keskittäen osaamisensa energia-alalle, kunnes 1990-luvun lama ohjasi yrityksen keskittämään toimintansa ydinosaamiseensa eli energiasektoriin sekä tulostus- ja käyttöpalveluihin. Vuonna 2001 Tietosavon nimeksi vaihtui Enfo, kun Kuopion Puhelin Oyj:sta tuli Tietosavon suurin omistaja. Saman vuosikymmenen puolivälissä Enfo laajensi IT-toimintaansa Ruotsiin ja Enfon ostamasta ruotsalaisesta IT-yritys Zpiderista muodostuu Enfo Sweden Ab. Se kasvoi 2010-luvulla ostamalla muun muassa ruotsalaisia bisnes- ja konsultointialan yrityksiä ja Enfon silloinen työntekijämäärä nousi lähes sadalla. Vuonna 2014 Enfo täytti 50 vuotta ja työllisti Suomessa ja Ruotsissa yhteensä 800 IT-alan ammattilaista. Yrityksen liikevaihto oli tuolloin lähes 160 miljoonaa euroa. (Enfoway 2014).

Enfo Oyj:n Service Desk toimii pääosin Kuopiossa, mutta osa henkilöstöstä on sijoitettuna myös Puolaan. Service Desk perustettiin 1990-luvun loppupuolella antamaan asiakkailleen käyttötukea muun muassa Microsoft Officen ohjelmiin. Siinä missä tukitoiminnot ovat sittemmin laajentuneet, on myös Service Deskin henkilöstön määrä noussut alkuaikojen muutamasta työntekijästä nykyiselle tasolle, joka on Suomen ja Puolan työntekijät yhteenlaskettuna noin 60 henkilöä. Kesällä 2013 Service Desk aloitti myös ympärivuorokautisen tukipalvelun. (Vähäkainu 2015-02-24.)

Service Desk on jaettu palvelutiimeihin, jotka palvelevat niille nimikoituja asiakkaita asiakasrajapinnassa puhelimitse, sähköpostitse tai chatin välityksellä. Jokaisella palvelutiimillä on oma esimiehensä.

Enfo Oyj:n Service Deskin saavutuksista tärkeimmät ovat:

- ISO 9001:2008 -laatusertifikaatti vuonna 2010 (Enfo 2010)
- Suomen paras Help Desk -palvelu vuonna 2011 (Enfo 2011)
- Suomen paras Service Desk -esimies vuonna 2012 (Enfo 2012)
- vuoden 2013 paras asiakaspalvelija, ”Asiakkaan Ääni” (Enfo 2013)
- vuoden 2014 paras Service Desk -asiantuntija (Enfo 2014).

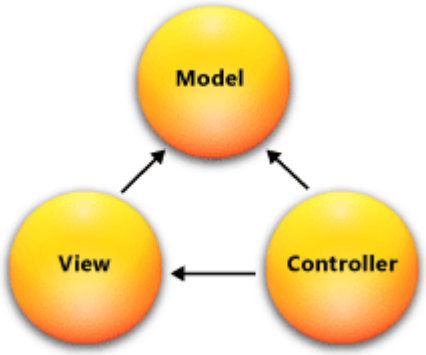
3 KÄYTETYT TEKNIIKAT JA TYÖKALUT

Työssä käytetyt ohjelmointitekniikat ja -työkalut valittiin sen vuoksi, että niitä käytettiin myös portaalin kehityksessä ja niistä oli jo entuudestaan kokemusta. Tässä luvussa on kerrottu lyhyesti käytettyjen tekniikoiden perusteista.

3.1 ASP.NET MVC ja Entity Framework

ASP.NET MVC (Model-View-Controller) on osa ASP.NET-ohjelmistokehystä, joka pohjautuu MVC-arkkitehtuuriin (kuva 1). MVC koostuu kolmesta osa-alueesta: mallista, näkymästä ja kontrollerista. MVC:stä on etua dynaamisten sivustojen luomisessa, ja koska esimerkiksi bisnes- ja käyttöliittymälogiikat voidaan erottaa toisistaan, on myös työtehtävien erottelu tiimin kesken helpompaa. Näkymä sisältää muun muassa käyttäjärajapinnan ja käyttöliittymälogiikan. Kontrolleri välittää tarvittavat tiedot näkymälle, prosessoi sieltä palautetut tiedot ja toimii yhteistyössä mallin kanssa. Mallia hyväksikäyttäen voidaan hakea tietoa tietokannasta ja palauttaa muuttunut tieto takaisin. (ASP.NET MVC Overview 2015.)

ASP.NET MVC 1.0 julkaistiin vuonna 2009, jonka jälkeen versioita on tullut tasaiseen tahtiin (ASP.NET MVC Framework Release History 2015). Portaali toteutettiin MVC 4 -versiolla, joten se oli luonnollinen valinta myös tähän työhön. MVC 5 olisi ollut myös vaihtoehto, mutta työssä päätettiin pysyä tutussa versiossa.



KUVA 1. MVC-kehitysmalli (ASP.NET MVC Overview 2015.)

Entity Framework on C#-pohjainen .NET-ohjelmistokehyksen työkalu, joka oleellisesti auttaa sovelluskehittäjää tietokannan tietojen käsittelyssä. Entity Framework sisältää uuden sovelluksen kehittämiseen kolme erilaista lähestymistapaa, jotka ovat Code First, Model First ja Database First. Tässä työssä käytettiin Code First -kehitysmallia, joka esiteltiin Entity Frameworkin 4.1-versiossa. Code First -kehitysmallin mukaan sovelluksessa käytettävät luokat luodaan manuaalisesti ilman erillistä visuaalista työkalua ja niiden perusteella luodaan kokonaan uusi tietokanta tai päivitetään jo olemassa olevaa tietokantaa. (Entity Framework Tutorial 2015.)

3.2 C#

C# on .NET-ohjelmistokehyksen C-pohjainen ohjelmointikieli, joka julkaistiin vuonna 2000. C# suunniteltiin yksinkertaiseksi ja nykyaikaiseksi, oliopohjaiseksi ohjelmointikieleksi, jota voidaan käyttää monenlaisissa kehitysympäristöissä. Se on pyrkinyt yhdistämään useiden ohjelmointikielten parhaita ominaisuuksia. (ECMA International 2006.)

3.3 HTML

HTML (Hyper-Text Markup Language) on hypertekstin merkintäkieli, jota käytetään laajasti muun muassa visuaalisten www-sivujen rakenteen määrittelyyn. HTML koostuu tageista, joiden perusteella internetselain muodostaa sivun ulkonäön ja linkitykset. Ensimmäinen HTML:n versio julkaistiin jo vuonna 1991. Uusin versio on HTML5, joka ilmestyi vuonna 2014. (W3Schools HTML 2015.)

3.4 JavaScript ja jQuery

JavaScript (ts. ECMAScript), jota ei tule sekoittaa Java-ohjelmointikieleen, on hyvin yleisesti käytössä oleva ohjelmointikieli. JavaScript keksittiin vuonna 1995 ja standardoitiin ensimmäisen kerran vuonna 1997. JavaScript toimii käyttäjän selaimella, ja sen avulla voidaan tehdä esimerkiksi tiedon tarkistuksia, luoda dynaamisuutta ja käytettävyyttä sekä muuttaa vaikkapa sivun HTML- tai CSS-muotoiluja. (W3Schools JS 2015.)

JavaScriptiin pohjautuva jQuery on monipuolinen ja kevytkäyttöinen JavaScript-kirjasto, ja sen avulla sivustoille voidaan helposti luoda runsaasti toiminnallisuuksia, kuten animaatioita ja erilaisia tapahtumakäsittelijöitä (W3Schools jQuery 2015).

3.5 Google Charts

Google Charts on eräs visuaalinen tapa esittää numeerista dataa www-sivuilla. Google Charts käyttää tiedonesitykseen HTML5-tekniikkaa ja SVG:tä, ja se on upotettavissa sivuun JavaScriptillä. Google Charts ei tarvitse toimiakseen erikseen asennettavia selainliitännäisiä, joten graafinen data on nähtävillä millä tahansa modernilla internetselaimella myös mobiilina (Using Google Charts 2015).

Työssä käytetty Line Chart -diagrammimalli prosessoi saamansa JSON-datan paikallisesti käyttäjän työasemalla, eikä se luovuta dataa ulkopuolisille palvelimille. Google Chartsilla on Creative Commons Attribute 3.0 -käyttölupa (http://creativecommons.org/licenses/by/3.0/), ellei kyseisen Google Chartin sivulla toisin mainita. Google Chartsin sivuilla esiintyvillä koodikatkelmilla on Apache 2.0 -käyttölupa (http://www.apache.org/licenses/LICENSE-2.0). (Google Line Chart 2015.)

3.6 JSON

JSON (JavaScript Object Notation) on kevyt tiedonsiirtotapa, joka toimii kaikkien ohjelmointikielten välillä. JSON on tekstijono, joka koostuu haka- ja aaltosulkeista, kaksoispisteistä ja pilkuista. JSON-

tiedosto on yksinkertainen muodostaa ja parsia, ja se on varsin helppolukuista tietokoneen lisäksi myös ihmiselle. (ECMA International 2013.)

3.7 Microsoft Visual Studio

Työssä käytettiin Microsoft Visual Studio Express 2013 for Web -ohjelmaa. Express-versio on lisenssimaksuton kehitystyökalu myös kaupallisten web-sovellusten kehittämiseen (Visual Studio 2015).

Visual Studio oli luonnollinen ratkaisu toteutustyökaluksi, koska se on ASP.NET MVC:n virallinen kehitystyökalu (Visual Studio Application Development 2015). Myös tietokannan käsittely onnistui Visual Studion avulla, joten muita ohjelmointityökaluja ei työn aikana tarvittu.

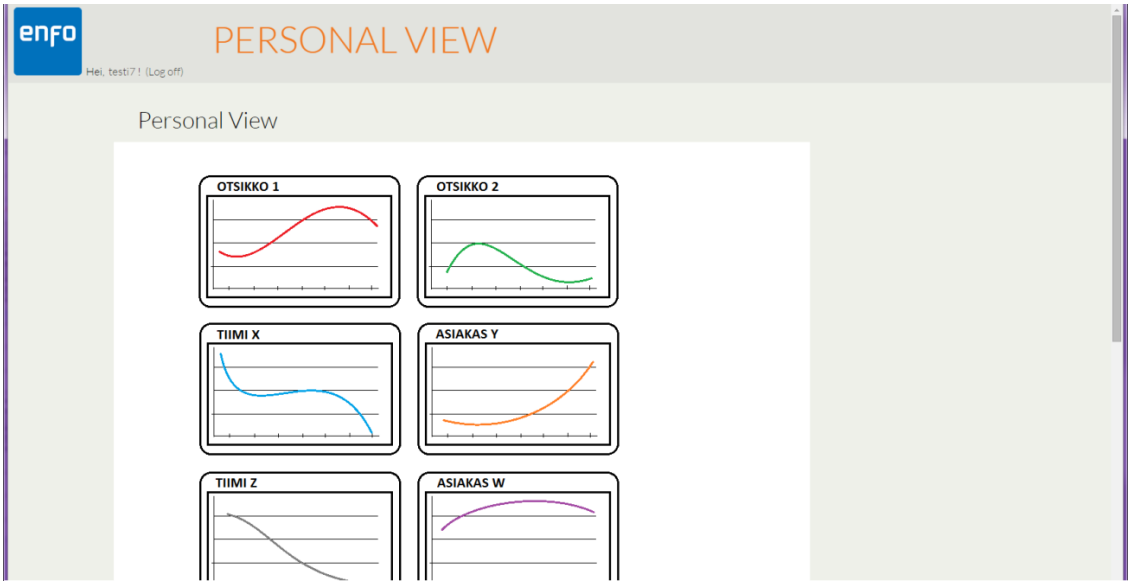
4 TYÖVAIHEET

Työssä päätettiin edetä niin kutsutun ketterän mallin mukaisesti, joka sopii lyhytaikaiseen projektiin, jonka osallistujamäärä on pieni. Etenemistä tarkasteltiin kahden viikon jaksoissa ja seuraavan jakson tavoitteet asetettiin päättyneen jakson aikaansaannosten mukaan.

4.1 Määrittely ja vaatimukset

Työn vaatimukset käytiin toimeksiantajan kanssa pääosin läpi jo opinnäytetyön aihetta pohdittaessa. Todettiin, että portaalin raporttien yksittäinen selaaminen on aikaa vievää, joten jokin yhteenvetonäkymä olisi tarpeellinen. Lähdekoodin tulisi olla helposti päivitettävissä ja mahdollisesti hyödynnettävissä myös muihin kehityskohteisiin portaalin myöhemmissä kehitysvaiheissa. Luonnollisia vaatimuksia työn lopputulokselle olivat esitettävän tiedon oikeellisuus, helppo käytettävyys ilman erillisiä käyttöohjeita ja virheellisen käytön estäminen.

Yksinkertainen sivunäkymä sisältäisi pienempiä komponentteja, joihin käyttäjä saisi itse määritellä näkyviin haluamansa graafit. Ajatuksena oli myös, että komponentteja olisi mahdollista myös järjestellä keskenään. Kuvassa 2 on esitetty sivunäkymän karkea ulkoasumäärittely, jossa ei ole vielä otettu kantaa toteutustapaan.



KUVA 2. Visio tulevan graafinäkymän rakenteesta

4.2 Suunnittelu

Jotta graafinäkymä saataisiin jokaiselle käyttäjälle henkilökohtaiseksi, täytyisi graafikomponenttien sisältämät tiedot tallentaa pysyvämmin johonkin ja tietojen tulisi olla helposti päivitettävissä. Alun suunnitteluvaiheessa pohdittiin parasta tallennuskeinoa komponenttien tiedoille.

Tallennusvaihtoehdoiksi nousivat joko työasemalle paikallisesti tallennettava eväste tai portaalin käytössä oleva tietokanta. Evästeen etu olisi ollut se, ettei sitä varten olisi tarvittu käyttää erillistä autentikointia eli käyttäjän tunnistamista, mutta sen heikkous olisi asetusten katoaminen evästeiden poiston tai työaseman vaihtumisen myötä. Varmempi asetusten pysyvyys tuntui suotuisammalta ratkaisulta, joten sen vuoksi evästeestä luovuttiin ja päädyttiin käyttäjän autentikointiin. Komponenttien asetukset voitaisiin tällöin tallentaa portaalin käyttämään tietokantaan, josta ne saataisiin helposti käyttöön sisäänkirjautuneen käyttäjän id:n perusteella.

Käyttäjän autentikointiin olisi myös ollut vaihtoehtoja aina AD:sta (Active Directory) alkaen. AD-autentikointi olisi luultavasti ollut loppukäyttäjän kannalta helpoin vaihtoehto, muttei ollut varmaa, miten helposti yrityksen AD-tietokantaa olisi päästy hyödyntämään. Erillisen käyttäjätunnuksen heikkous olisi puolestaan se, että graafinäkymään päästäkseen käyttäjä tarvitsisi nykypäivän monien käyttäjätunnusten ja salasanojen lisäksi vielä yhdet tunnukset, ja ne saattaisivat unohtua. Työn toteutuspainotteisen tavoitteen vuoksi käyttäjätunnistus oli silti aiheellisinta ja yksinkertaisinta toteuttaa ASP.NET MVC:n omalla WebSecurity-autentikoinnilla. Sen avulla sisäänkirjautuneen käyttäjän tiedot voitiin noutaa tietokannasta melko vaivatta ja päästiin nopeasti etenemään itse työhön.

4.3 Tekninen toteutus

Työn alkuvaiheessa sen tavoitteet olivat vielä melko avoimet. Suunnittelu oli sen vuoksi aluksi hyvin pääpiirteittäistä ja se pyrittiin tekemään nopeasti. Ketterän mallin mukaisesti myös seuraavat jaksot sisälsivät toteutuksen lisäksi hieman suunnitteluakin. Suunnittelusta siirryttiin kaikissa jaksoissa varsin nopeasti toteutukseen, koska työhön käytettävä aika oli rajallinen. Ketteryyden ansiosta toteutusmenetelmien vaihto oli mahdollista, jos aiottu menetelmä ei olisi jostain syystä toiminutkaan. Sopivimmat ja optimaalisimmat menetelmät löytyivät sujuvasti työskentelyn aikana, kun tavoite työn lopputuloksesta oli selkeä.

Työ toteutettiin toimeksiantajan Visual Studio -projektiin, joka sisälsi portaalin lähdekoodin ja josta otettiin työtä varten täydellinen kopio. Työn toteutus koostui pohjatyöstä (käyttäjän autentikointi, skriptipakettien päivitykset, mallin ja tietokannan muokkaus), ohjelmoinnista (ulkoasun muotoilu, sisällöntuotanto) ja käytettävyyden optimoinnista. Lähdekoodin toimivuutta testattiin runsaasti koko työn ajan.

4.3.1 Tietokanta, malli ja migraatiot

Portaalin Microsoft SQL-tietokanta sisältää kaikki opinnäytetyöhön tarvitut tiedot. Heti työn alussa portaalin kannasta luotiin kopio, testitietokanta, jonka avulla uusia toimintoja sai kokeilla ilman, että häirittiin varsinaisen tuotantotietokannan toimintaa. Testitietokantaan lisättiin työn aikana graafinäkymän komponenttien tiedoille oma taulu, asetustaulu. Työssä tarvitut ja muokatut tietokanta-osiot rajoittuvat asetustauluun, UserProfiles-tauluun ja projektin generoimiin käyttäjäroolitiedot sisältäviin webpages_Roles ja webpages_UsersInRoles -tauluihin. Muihin tietokannan osioihin ei tarvittu työn aikana kajota, niistä haettiin vain tarvittavaa tietoa.

Testitietokanta luotiin muokkaamalla projektin juuressa sijaitsevassa web.config-tiedostossa olevaa osoitepolkua siten, että tietokannan nimeksi muutettiin mikä tahansa muu nimike (esimerkkinä kuvassa 3 sinisellä korostettuna ”testikanta”). Kun projekti ajettiin, osoitepolussa määritellylle palvelimelle muodostui automaattisesti uusi senhetkisen mallin mukainen tietokanta. Tällä tavoin muodostettu tietokanta on kuitenkin sisällöltään tyhjä, joten sinne siirrettiin tuotantotietokannasta hieman dataa testausta varten.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <configSections>
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFr
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237
  <sectionGroup name="dotNetOpenAuth" type="DotNetOpenAuth.Configuration.DotNetOpenAuthSection, DotNetOpenAuth.Co
      <section name="messaging" type="DotNetOpenAuth.Configuration.MessagingElement, DotNetOpenAuth.Core" require
      <section name="reporting" type="DotNetOpenAuth.Configuration.ReportingElement, DotNetOpenAuth.Core" require
    <section name="openid" type="DotNetOpenAuth.Configuration.OpenIdElement, DotNetOpenAuth.OpenId" requirePermis
  <connectionStrings>
    <add name="xxxxDBContext" connectionString="Data Source=0.0.0.0\KannanOsoite;Initial Catalog=testikanta;Integ
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="2.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="PreserveLoginUrl" value="true" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>
```

KUVA 3. Esimerkki uuden tietokannan luonnista web.config-tiedoston osoitepolkua muuttamalla

Testitietokantaan lisättiin graafinäkymän asetuksille oma taulu, asetustaulu. Se tehtiin Code First -toimintatavan mukaisesti luomalla Visual Studion projektiin ensin graafinäkymän malli, jonka lisäksi projektin Models-kansiossa olevaan DBContext-tiedostoon lisättiin uusi tietokantaviite eli DbSet. Kuvassa 4 on esimerkkikoodi uuden mallin ja tietokantaviitteen määrittämisestä. Mallin luomisen jälkeen testitietokanta voitiin päivittää mallia vastaavaksi ja asetustaulu muodostui muiden taulujen joukkoon.

```
// esimerkki contextin ja uuden tietomallin luomisesta
public class EsimContext : DbContext
{
    public EsimContext()
        : base("EsimConnection")
    {
    }
    public DbSet<Esimerkki> Esimerkki { get; set; }
}

[Table("Esim")]
public class Esimerkki
{
    [Key]
    public int ID { get; set; }

    [Required]
    [Display(Name = "Ensimmäinen sarake")]
    public string sarake1 { get; set; }

    [Display(Name = "Toinen sarake")]
    public string sarake2 { get; set; }
}
```

KUVA 4. Esimerkki uuden mallin luomisesta ja DbSetin lisäämisestä

Malliin lisättiin myös muihin tauluihin osoittavia virtuaalisia viittauksia, jotka mahdollistavat Lazy Loading -tiedonhaun viitatuista tauluista (MSDN Lazy Loading 2015). Viittausten luonti aiheutti testitietokannan muodostuksessa sen, että jokaiseen edellämainitulla tavalla viitattuun tauluun generoitui vierasavain, joka osoitti tuohon uuteen tauluun. Vierasavaimet osoittautuivat työn kannalta turhiksi, mutta ne päätettiin jättää työn ajaksi testiympäristöön, koska niiden poistamisesta olisi saattanut koitua ylimääräistä ajankulua.

Graafinäkymän asetustauluun tarvittiin tehdä työn aikana muutoksia, koska työn alussa ei vielä tiedetty kaikkia tarvittavia sarakkeita. Muutokset tehtiin ensin malliin, josta ne ajettiin tietokantaan Code First -migraatioilla Visual Studion NuGet Package Manageria käyttäen. Jotta migraatioita pystyttiin tekemään, täytyi niiden käyttö ensin sallia Enable-Migrations -komennolla, jonka jälkeen migraatiot voitiin asettaa myös automaattisiksi. Kun malliin tehtiin muutoksia, ne päivitettiin tietokantaan Update-Database -komennolla. NuGet Package Manager antoi lisäohjeita, jos oli olemassa useampia kohteita joihin migraatiot haluttiin sallia tai taulun poistettavat sarakkeet sisälsivät jo tietoa. Jos Visual Studion Server Explorer -välilehden päivitys ei tuonut tehtyjä tietokantamuutoksia näkyviin, Visual Studio jouduttiin sulkemaan ja käynnistämään uudelleen.

4.3.2 Projektin skriptipakettien päivitys

Pohjaprojektissa käytettävät skripti- ja referenssipaketit päivitettiin niiden uusimpiin versioihin Visual Studion NuGet Package Managerilla. Update-Package -komento päivittää projektin kaikki ne paketit, joista löytyy uusi versio. Pakettien päivityksen jälkeen ohjelmaa ajettaessa ilmaantui virhe, joka kehotti päivittämään Razorin uuden versionumeron web.config-tiedostoon (kuva 5), joka sijaitsee projektin juuressa. On huomioitava, että toinen samanniminen tiedosto sijaitsee Views-kansion juuressa, mutta sitä ei tarvita käsitellä tässä työssä.

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="WebMatrix.Data" publicKeyToken="31bf3856ad364e35" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-2.0.0.0" newVersion="2.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.WebPages.Razor" publicKeyToken="31bf3856ad364e35" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-2.0.0.0" newVersion="3.0.0.0" /> <!---Updated from 2.0.0.0, 19.2.2015-->
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="DotNetOpenAuth.AspNet" publicKeyToken="2780ccd10d57b246" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-4.3.0.0" newVersion="4.3.0.0" />
    </dependentAssembly>
```

KUVA 5. Razorin versionumeron päivitys web.config -tiedostoon

4.3.3 Kontrollerit

Kun portaalin Visual Studio -projekti aikanaan luotiin, siihen generoitui automaattisesti käyttäjän tunnistukseen tarvittavat mallit, näkymät ja kontrolleri. Työssä käsiteltiin kaikkiaan kolmea kontrolleria (controller), jotka sijaitsevat projektin Controller-kansiossa. Yksi niistä on projektiin generoitunut käyttäjätunnusten hallintaan tarkoitettu kontrolleri, johon tehtiin pieniä muutoksia muun muassa käyttäjän edelleenohjausta ja käyttäjäroolitusta varten. Siellä myös muodostettiin asetustauluun tallennettavat käyttäjäkohtaiset rivit jo käyttäjän rekisteröitymisvaiheessa. Portaalin

kontrolleriin lisättiin käyttäjätunnusten ja -roolien käsittely ja poistot. Itse graafinäkymään liittyvässä kontrollerissa käsiteltiin vain graafinäkymän ja sen osanäkymien tietoja sekä hallittiin graafien JSON-tiedonhaku.

4.3.4 Käyttäjäroolit ja käyttäjän autentikointi

Sisäänkirjautumisen ja rekisteröitymisen näkymät muokattiin portaaliin soveltuviksi. Autentikointiin liittyvät taulut generoituivat testitietokantaan ensimmäisen käyttäjän rekisteröinnin yhteydessä, jonka jälkeen käyttäjäroolit sisältävään webpages_Roles-tauluun lisättiin uudet käyttäjäroolit. Pääkäyttäjän käyttäjärooli määritettiin tietokantaan manuaalisesti, mutta sen jälkeen luotavien käyttäjätunnuksen perusroolimääritys ohjelmoitiin tapahtumaan automaattisesti jo rekisteröinnin yhteydessä. Erillisestä pyynnöstä esimerkiksi esimiehen käyttäjätunnukselle voidaan myöhemmin myöntää korkeampi käyttäjärooli.

Edellisiä toimintoja haluttiin testata luomalla uusi testitietokanta kohdassa 4.3.1 kerrotulla tavalla. Uuteen testikantaan käyttäjätunnuksia luotaessa tapahtui virhe, koska ensimmäistä käyttäjää rekisteröitäessä ei ollut webpages_Roles-taulussa vielä sisältöä käyttäjäroolin myöntämisvaiheessa. Tällöin saatiin esille virheilmoitus, ettei tarvittua käyttäjäroolia ole, mutta käyttäjätunnus tuli siitä huolimatta luoduksi. Kontrollerin rekisteröitymiskohtaan lisättiin sen vuoksi tarkistus, että onko haluttu käyttäjärooli jo olemassa. Jos kyseessä on ensimmäisen käyttäjätunnuksen luonti eikä tarvittavia käyttäjärooleja vielä ole, ne luodaan webpages_Roles-tauluun automaattisesti. Sen ansiosta uusien testitietokantojen luonnissa ei koitunut enää ongelmia.

Asetustauluun ja webpages_UsersInRoles-tauluun lisättiin Visual Studiota käyttäen vierasavaimet, jotka muodostivat relaation niiden ja käyttäjän autentikointiin liittyvän UserProfiles-taulun välille. Relaation vuoksi käyttäjätunnusten poisto ei onnistunut, koska asetus- ja webpages_UsersInRoles -tauluihin jäi käyttäjätunnukseen viittaavia rivejä. Sen vuoksi vierasavaimiin määritettiin komento, joka poistaa relaatiotaulujen rivit samalla kun käyttäjätunnus poistetaan. Sen jälkeen käyttäjätunnuksen poisto onnistui ongelmitta ja myös relaatiotaulujen tiedot poistuivat ilman konflikteja.

Tietyt portaalin sivut haluttiin asettaa autentikoinnin piiriin. Kaikille saman kontrollerin hallinnoimille sivuille pääsy oli helppo rajata lisäämällä suoraan kontrollerin alkuun attribuutti [Authorize]. Myös yksittäisten sivujen rajaus olisi ollut mahdollista, mutta se ei ollut tarpeellista. Sen sijaan eräille sivulle pääsy haluttiin rajata vain tietyille käyttäjärooleille, mikä tehtiin lisäämällä halutut roolit [Authorize]-attribuutin sisään.

Tietokantaan lisättiin UserProfiles-tauluun oma sarake käyttäjän viimeisimmälle sisäänkirjautumisajalle, joka määritettiin tallentumaan aina sisäänkirjautumisen yhteydessä (kuva 6). Sisäänkirjautumisajan tallennus helpottaa käyttäjätunnusten hallinnointia ja graafinäkymän käyttäjäasteen seuraamista. Käyttäjätunnusten hallinnoinnin varsinaisesta näkymästä on kerrottu tarkemmin kohdassa 4.3.7.

```
using (UsersContext db = new UsersContext())
{
    UserProfile UProf = db.UserProfiles.SingleOrDefault(x => x.UserName == model.UserName);
    UProf.LastLogin = DateTime.Now;
    db.Entry(UProf).State = EntityState.Modified;
    db.SaveChanges();
}
```

KUVA 6. Käyttäjän viimeisimmän sisäänkirjautumisajan tallennus tietokantaan

Rekisteröityneen ja sisäänkirjautuneen käyttäjän id:tä tarvittiin eräissä toiminnallisuuksissa. Kuvassa 7 näkyy kolme id:n hakemistapaa, joita kokeiltiin rekisteröitymisen yhteydessä. Lähdekoodin testauksessa palautuneet arvot näkyvät kuvassa 7 kommentoituna. Id:n nouto oli ensimmäisen kerran ajankohtaista jo rekisteröitymisvaiheessa, koska sen yhteydessä luotiin graafinäkymän asetustauluun kyseiselle käyttäjätunnukselle valmiiksi määritellyt rivit itse graafinäkymää varten.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult [redacted](RegisterModel model, string valid)
{
    if (ModelState.IsValid && [redacted])
    {
        // Attempt to register the user
        try
        {
            WebSecurity.CreateUserAndAccount(model.UserName, model.Password);
            if(WebSecurity.HasUserId)
            {
                WebSecurity.Logout();
            }
            WebSecurity.Login(model.UserName, model.Password);
            Roles.AddUserToRole(model.UserName, "[redacted]");
            // int id = (int)Membership.GetUser().ProviderUserKey;  // palautti arvon 0
            // int id = (int)WebSecurity.CurrentUserId;             // palautti arvon -1
            int id = WebSecurity.GetUserId(model.UserName);         // palautti oikean id-arvon
```

KUVA 7. Rekisteröityneen käyttäjän id:n haku rekisteröitymisen yhteydessä

4.3.5 Näkymien ulkoasu ja muotoilu

MVC:ssä näkymien ulkoasupohja muodostetaan omassa tiedostossaan, joka on ASP.NET MVC -projekteissa oletuksena _Layout.cshtml. MVC:n ulkoasutiedosto sisältää viittaukset tarvittaviin CSS-, JavaScript- ja jQuery-tiedostoihin sekä HTML-koodia, jolla määritetään HTML-sivun alku ja loppu sekä meta-, header- ja body-tiedot. Mikäli jossain näkymässä halutaan käyttää jotain muuta kuin oletustiedostoa, täytyy se määrittää näkymän lähdekoodissa. Graafinäkymälle ja hallintaosiolle luotiin omat ulkoasutiedostonsa, koska niiden pääotsikoinnit poikkeavat muusta portaalista. Kuvassa 8 on esimerkkikoodi ulkoasutiedoston perusrakenteesta. RenderBody-funktio muodostaa sivun sisällön, joka määritetään kussakin näkymässä.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta http-equiv="refresh" content="360" />
        <meta charset="utf-8" />
        <title>Raporttiyhteenveto - @ViewBag.user</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />

        <meta name="viewport" content="width=device-width" />
        <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
        @Scripts.Render("~/bundles/jquery")
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
        @Scripts.Render("~/bundles/jqueryui")

    </head>
    <body onload="checkFirstVisit()">
        <header>
            <!-- sivun otsikon, logon ja sisäänkirjautumisen divit -->

        </header>

        <div id="body">
            @Scripts.Render("~/bundles/jquery")
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
            </div>

        </footer>
    </body>
</html>
```

KUVA 8. Esimerkki _Layout-tiedoston perusrakenteesta

Projektin yhteiseen CSS-tiedostoon luotiin omat muotoilunsa graafinäkymää varten, koska div-komponenttien kohdistukset poikkeavat muista näkymistä. Graafinäkymän sisältämien komponenttien ulkoasu puolestaan määräytyy pääosin jQuery UI:n skriptien mukaan. Osa skriptitiedostoista on Visual Studion projektissa itsessään, mutta koska osa komponentin olennaisista toiminnoista jäi siitä huolimatta vajaiksi, on projektissa myös viittauksia jQueryn lähdepalvelimelle.

4.3.6 Graafiosion näkymät ja käytettävyys

Graafiosion näkymiä (view) on kolme, joista yksi on varsinainen päänäkymä ja loput kaksi ovat osanäkymiä (partial view). Käyttäjätunnusten hallinnointia varten luotiin oma näkymä, josta kerrotaan tarkemmin kohdassa 4.3.7.

Päänäkymän sisäkkäistä div-rakennetta on havainnollistettu kuvassa 9. Siinä on yksi suuri div-alue (rajattu kuvassa sinisellä), jonka sisällä olevat div-komponentit (yksi esimerkki rajattu kuvassa punaisella) muodostettiin dynaamisesti sisäkkäisissä foreach-silmukoissa. Niissä käytiin läpi kontrollerilta saatu lista käyttäjän komponenttien sisällöstä ja tietokantaan tallennetusta id-järjestyksestä, jolloin graafit saatiin tulostettua oikeaan järjestykseen.

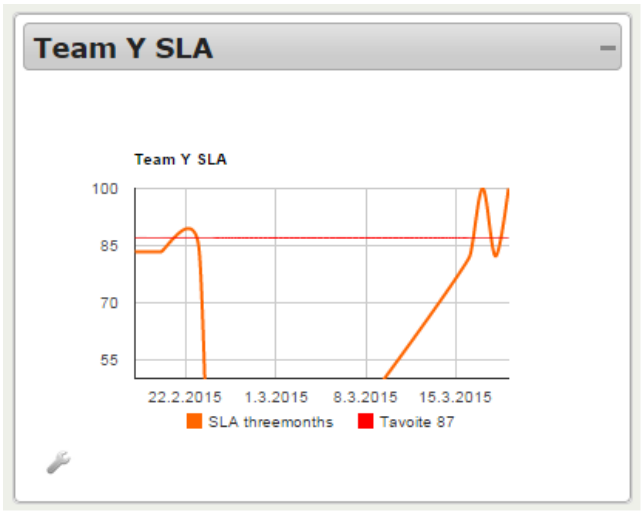
KUVA 9. Graafinäkymän rakenne

Perustoimintojen luomisen jälkeen työn pääpaino oli graafiosion helpossa käytettävyydessä ja selkeässä esitystavassa. Näkymässä ja osanäkymissä on käytetty runsaasti JavaScript- ja jQuery-koodia helpottamaan käytettävyyttä ja luomaan dynaamista kuvaa käyttäjälle. Käytettävyyden kulmakivet olivat käytön yksinkertaisuus ja helppous, käyttäjän manuaalisten syötteiden korvaaminen muilla keinoin sekä graafien selkeä tulkittavuus.

Päänäkymän komponenttien paikkoja voidaan vaihdella ja niiden id-järjestys voidaan tallentaa tietokantaan. Paikkojen tallennus perustuu jQuery:n sortable-widgetin sisältämään toArray-metodiin, joka muodostaa div-komponenttien id:eistä string-muotoisen listan, joka tallennetaan kantaan kyseisen käyttäjän tietoihin. Näkymä ehdottaa uuden järjestyksen tallentamista heti, kun komponenttien paikkojen muuttuminen on havaittu (sortablen update-tapahtuma). Kuvassa 10 punaisella rajattu tallennusehdotus toteutettiin sijoittamalla päänäkymän oikeaan alakulmaan yksinkertainen jQueryn hide-funktiolla piilotettu div-osio, joka kutsutaan näkyviin sortablen updaten tapahtuessa. Mikäli käyttäjä haluaa säilyttää uuden järjestyksen, post-metodi vie toArrayn muodostaman id-listan kontrollerille tietokantaan tallennettavaksi päänäkymää häiritsemättä. Mikäli tallennus ei onnistu, siitä annetaan erillinen ilmoitus. Käyttäjä voi myös olla säilyttämättä tekemiään muutoksia, jolloin komponentit palaavat alkuperäisille paikoilleen sivun päivityksen yhteydessä.

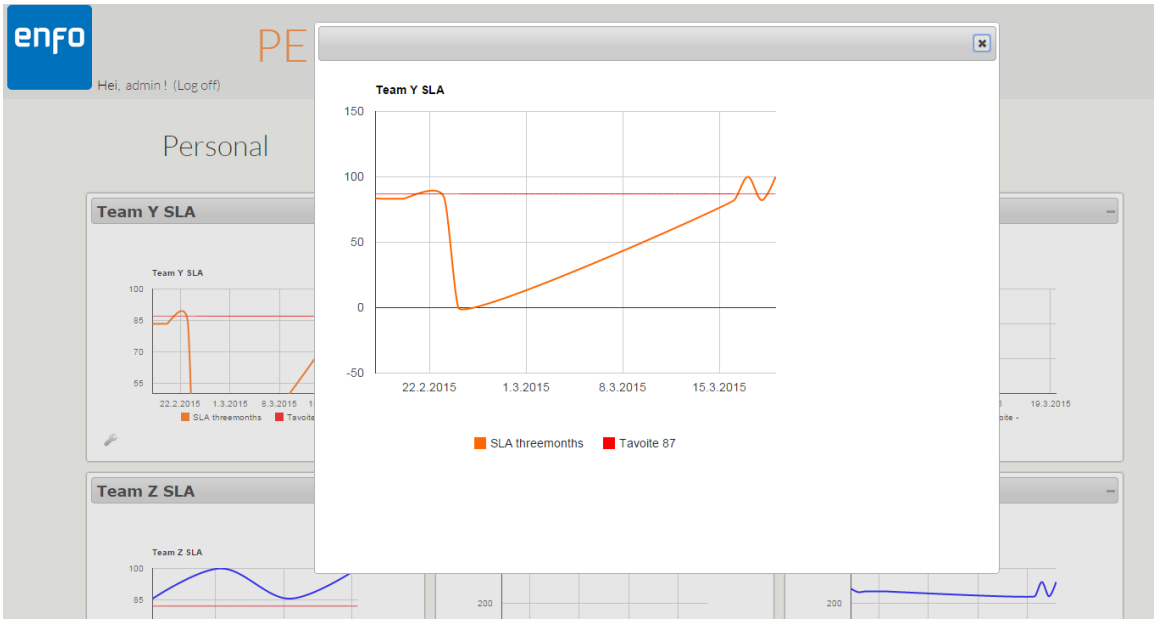
KUVA 10. Komponenttien järjestyksen tallentamista ehdottava div-osio

Päänäkymä sisältää kaksi erilaista osanäkymää, jotka avataan jQuery-dialogeihin. Dialogit ovat päänäkymässä div-osioina samalla tavoin kuin edellisessä kappaleessa mainittu tallennusehdotus, mutta niitä ei tarvitse erikseen piilottaa, koska ne ovat sisällöltään tyhjiä, kun dialogi on kiinni. Kunkin komponentin (kuvassa 11) sisältämä graafi ja jakoavaimen muotoinen asetuspainike toimivat HTML-linkkeinä, joihin on liitetty kyseisen komponentin id. Kumpaankin linkkiin on sidottu oma skriptinsä, joka avaa dialogin ja muodostaa kutsuttavan kontrollerin metodin nimestä ja linkin id:stä oman URL:nsa. Tällä tavoin kutsutulle kontrollerille saadaan välitettyä myös id, jonka perusteella kontrolleri hakee kannasta kyseisen komponentin tiedot. Kontrolleri palauttaa näin haetut tiedot osanäkymään, joka puolestaan muodostaa juuri avatun dialogin sisällön.



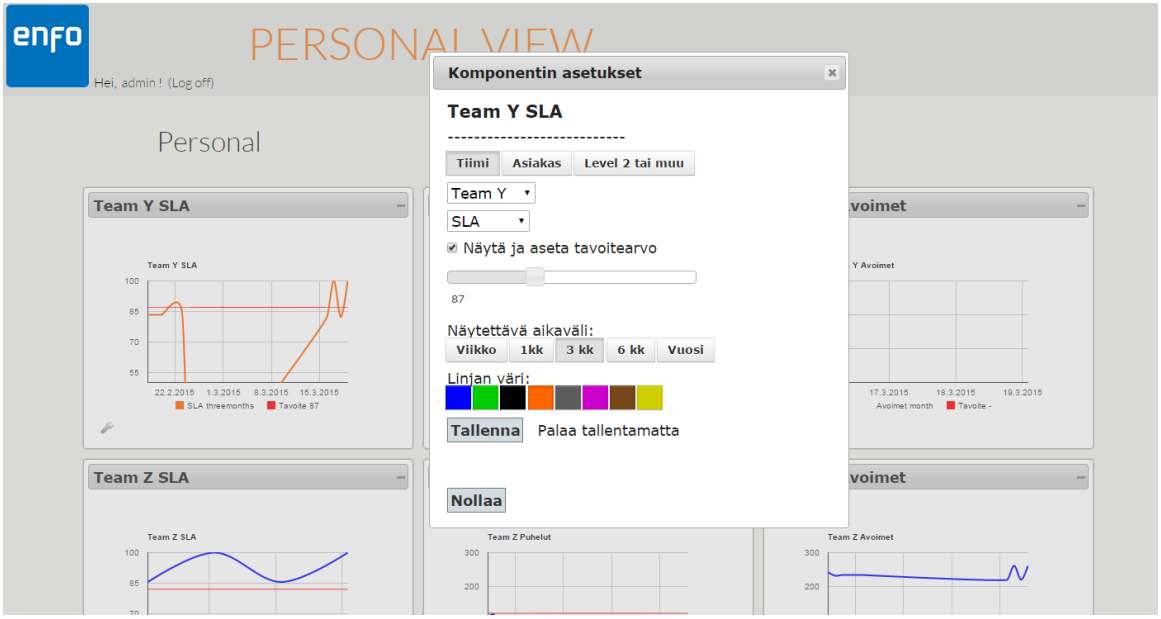
KUVA 11. Graafinäkymän komponentti

Graafin suurentava osanäkymä (kuva 12) luotiin, koska päänäkymässä päätettiin näyttää kahden sijasta kolme komponenttia rinnakkain, jolloin yksittäinen graafi on turhan pieni tarkempaan tarkasteluun. Osanäkymädialogi avautuu edellisessä kappaleessa mainitulla tavalla, ja dialogin graafi muodostetaan JavaScriptillä aivan kuten päänäkymässäkin, mutta ilman foreach-silmukoita. Dialogissa ei ole toiminnallisuutta, vaan siitä on ainoastaan paluu takaisin päänäkymään. Graafinäkymässä graafin pystyakselin minimi- ja maksimiarvot on määritetty staattisesti, jonka vuoksi arvojen ulkopuolinen viiva jää piiloon. Suurennusdialogissa graafi näkyy kokonaisuudessaan, koska siellä kyseisiä minimi- ja maksimiarvoja ei ole.



KUVA 12. Graafin suurentava osanäkymädialogi

Asetuspainikkeesta avautuu omaan dialogiinsa toinen osanäkymä (kuvassa 13), jossa käyttäjä voi määritellä, mitä tietoa hän haluaa kyseisessä komponentissa tarkastella. Käyttäjällä on dialogissa kolme vaihtoehtoa: tallentaa senhetkiset asetukset, nollata ne tai palata tallentamatta muutoksia. Kaikissa tapauksissa dialogista palataan takaisin päänäkymään, joka päivittyy valinnan jälkeen automaattisesti.



KUVA 13. Asetusdialogi

Tarkasteltavan kohteen ja aiheen valinta on toteutettu alasvetovalikoilla ja perinteisen radiobutton-valikoiman korvaavilla painikkeilla. Painikkeiden valinnan perusteella jQuerylla määritetään alasveto-valikoiden sisältö ja näkyvyys. Asetusdialogin otsikko, esimerkiksi kuvassa 13 näkyvä Team Y SLA, muodostetaan kontrollerissa painikkeiden ja alasvetovalikoiden valintojen mukaan, eikä käyttäjä voi sitä manuaalisesti muuttaa.

Käyttäjä voi määritellä graafiin tavoitearvon, joka erillisenä vaakaviivanaan selkeyttää ja nopeuttaa graafin tulkintaa. Arvon muokkaukseen valikoitui vetovalitsin (jQueryn slider), jota liu'uttamalla ta-voitearvo saadaan valittua. Tavoitearvon asettaminen on vapaaehtoista.

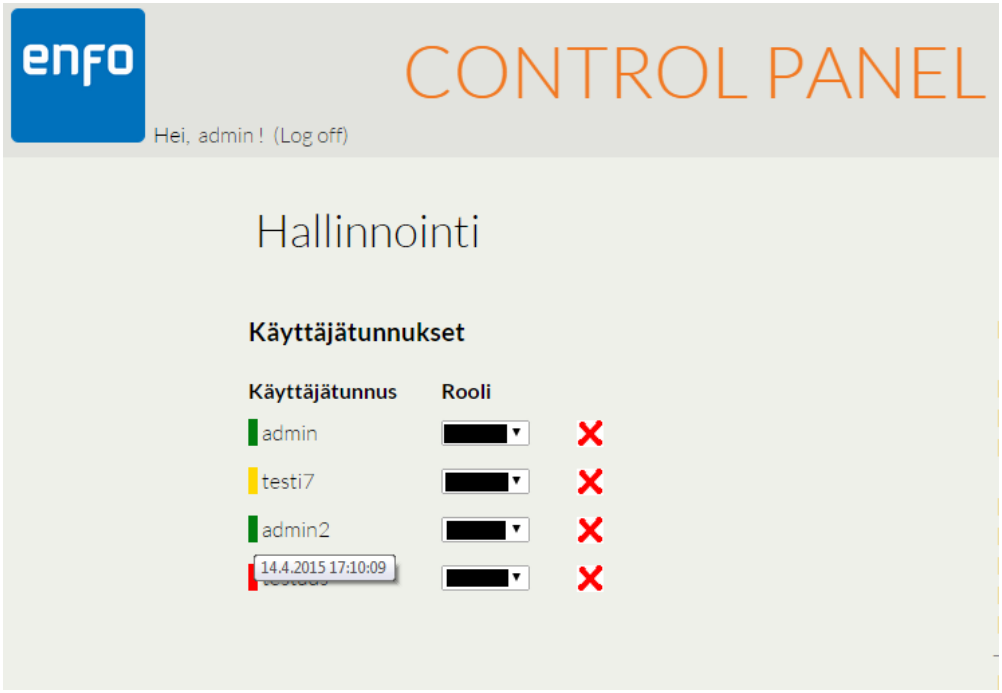
Päänäkymän monotonisuuden välttämiseksi käyttäjän on mahdollista valita graafien linjoihin erilaiset värit. Aluksi myös värivaihtoehdot olivat asetusdialogissa tavallisina radiobuttoneina, mutta visuaali-semman ilmeen vuoksi nekin vaihdettiin värillisiksi painikkeiksi. Valittu väri tallentuu tietokantaan string-muodossa heksadesimaalilukuna, esimerkiksi #FF6600 (RGB Color Values 2011).

Kun haettavan tiedon rajausta mietittiin, päätettiin antaa käyttäjälle mahdollisuus vaihtaa tarkastel-tavan aikavälin pituutta nykyhetkestä taaksepäin. Maksimivaihtoehdollakin graafit pysyvät selkeäm-pinä ja sivu latautuu nopeammin, kun haettavan tiedon määrä on rajattu.

Graafinäkymän asetusdialogin alasvetovalikoiden sisällön muodostukseen ja JSON-tiedonhakuun jäi staattisuutta, koska hakuvaihtoehtoja laajennettiin vasta työn loppupuolella. Muut asetusten toimin-nallisuudet oli jo tarkistettu ja optimoitu, mutta alasvetovalikon listan dynamisointi olisi vaatinut enemmän aikaa ja perehtymistä. Tässä vaiheessa kehitystyötä staattinen vaihtoehto osoittautui kui-tenkin dynaamista paremmaksi ja turvallisemmaksi. Staattisuus ei kuitenkaan ole este lähdekoodiin myöhemmin tehtäville muutoksille, mutta niissä on muistettava ottaa käsittelyyn kaikki muuttujat tai tietokannan sarakkeet, joihin uudelleennimetty muuttuja tai listan osa vaikuttaa.

4.3.7 Käyttäjätunnusten hallintanäkymä

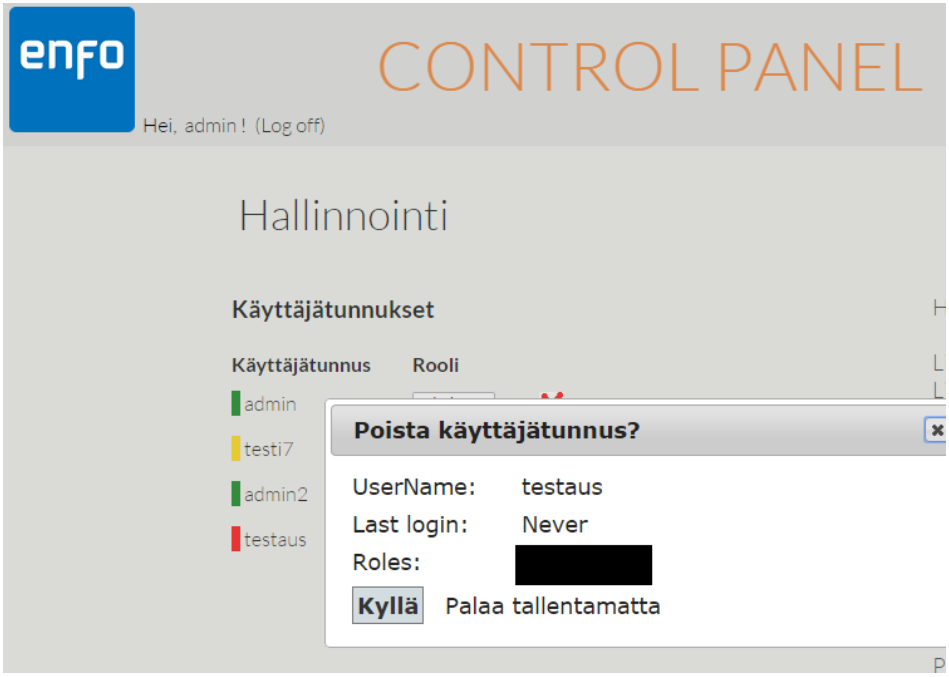
Portaalin hallintapaneeliin luotiin käyttäjätunnusten hallintanäkymä (kuva 14), jonka avulla portaalin pääkäyttäjä voi poistaa käyttäjätunnuksia tai muuttaa niiden käyttäjärooleja. Yksinkertainen näkymä listaa tietokannasta löytyvät käyttäjätunnukset, niiden käyttäjäroolit ja viimeisimmän sisäänkirjautumisajan. Jokaisen käyttäjätunnuksen yhteydessä on pieni väri-indikaattori, jonka väri vaihtuu sen mukaan, kuinka kauan viimeisimmästä sisäänkirjautumisesta on kulunut.



KUVA 14. Käyttäjätunnusten hallintanäkymä

Pääkäyttäjä voi helposti vaihtaa kunkin tunnuksen käyttäjäroolin alasvetovalikosta. Käyttäjäroolin vaihtaminen toteutettiin suoraan hallintanäkymään siten, että vaihtamista varten ei tarvita eikä avata erillistä osanäkymää, vaan vaihto tapahtuu kontrollerissa taustatyönä. Uusi rooli lähetetään post-metodina kontrollerille tietokantaan tallennettavaksi. Muutos ei normaalisti aiheuttaisi häiriötä näkymässä, joten tiedon tallennuksen yhteyteen lisättiin sivun päivitys, joka antaa käyttäjälle visuaalisen palautteen muutoksen tekemisestä. Jos tietokannassa on paljon käyttäjätunnuksia tai sivun päivitys käy jostain syystä muuten raskaaksi, voidaan päivitystoiminto ottaa pois käytöstä näkymän lähdekoodia muokkaamalla.

Käyttäjätunnuksen poisto tapahtuu käyttäjätunnuksen viereisestä rastista. Poiston vahvistus avautuu osanäkymädialogiin (kuva 15), joka avataan samalla tavoin kuin kohdassa 4.3.6, jossa on kerrottu graafiosion osanäkymien dialogeista. Dialogi kertaa käyttäjätunnuksen tiedot ja pyytää käyttäjää vahvistamaan poiston tai palaamaan takaisin. Valitun toiminnon jälkeen osanäkymästä palataan takaisin hallintanäkymään, joka päivittyy tarvittaessa vastaamaan tehtyjä muutoksia.



KUVA 15. Osanäkymädialogi käyttäjätunnuksen poiston vahvistuksesta

Käyttäjän salasanan resetointimahdollisuutta pohdittiin myös työn aikana, mutta se päätettiin jättää opinnäytetyön ulkopuolelle. Mikäli käyttäjä unohtaisi salasanan, se voisi olla resetoitavissa pääkäyttäjän toimesta. MVC:n WebSecurity sisältää metodit GeneratePassWordResetToken ja ResetPassword, joiden avulla valitun käyttäjätunnuksen salasana pystyttäisiin vaihtamaan uuteen. Ongelmaksi muodostuisi luultavasti se, että kuinka todentaa käyttäjän kertoma käyttäjätunnus, jos se ei vastaisi yrityksessä yhteisesti sovittua tunnuskäytäntöä.

4.3.8 Graafit ja JSON

Portaalin raporttidata esitetään visuaalisessa muodossa Google Charts -graafeilla. Portaalin kehitysvaiheessa Google Charts valikoitui käyttöön, koska se on helppokäyttöinen työkalu graafisten kuvaajien muodostukseen. Graafisen kuvaajan toteutukseen olisi ollut myös muita JavaScript-pohjaisia vaihtoehtoja, kuten Highcharts tai JS Charts, mutta työssä päätettiin pysyä tutussa ja yhtenevässä tekniikassa.

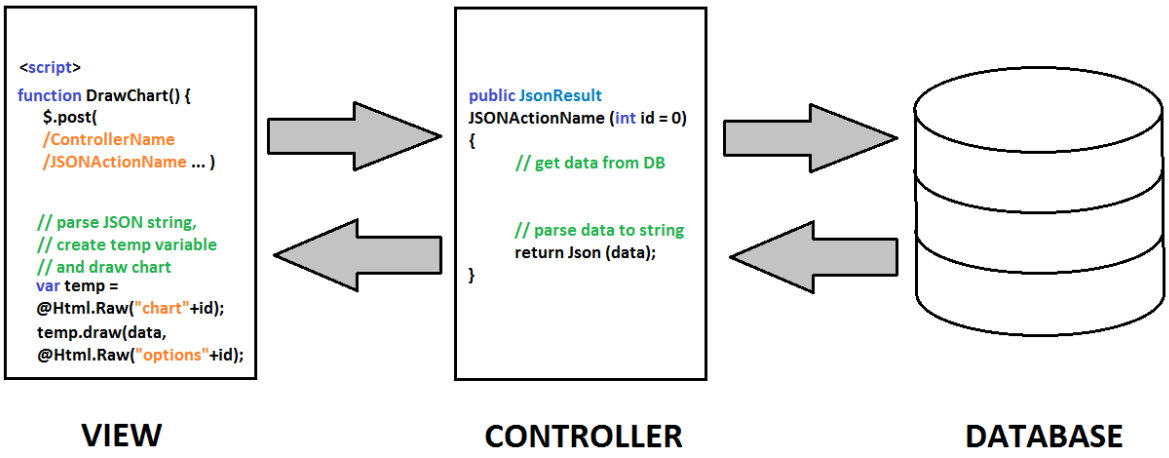
Kuten kohdassa 4.3.6 kerrotaan, graafiosion päänäkymän div-komponentit ja niiden sisältämät graafit luodaan sisäkkäisissä foreach-silmukoissa. Kunkin graafin kohdalla JavaScript kutsuu kontrollerissa sijaitsevaa JsonResult-metodia, joka noutaa sille välitetyn id:n perusteella kyseisen komponentin tiedot. Tiedot toimivat hakuparametreina, joilla tietokannasta haetaan varsinaista historia- ja raporttidataa. Metodi luo haetusta datasta JSON-muotoisen string-listan, joka palautetaan takaisin näkymän graafille. Näkymän JavaScript parsii JSON-listan graafiseksi kuvaajaksi. Kuvassa 16 on esimerkkikoodi näkymän JavaScriptistä ja kuviossa 1 on havainnollistettu näkymän, kontrollerin ja tietokannan välistä tapahtumaa.

```
@foreach (var item in ViewBag.items)
{
    <div class="div">
        <div class="divcontent">
            <script type="text/javascript" charset="utf-8">
                google.load("visualization", "1", { packages: ["corechart"] });
                google.setOnLoadCallback(drawChart);

                function drawChart() {
                    $.post('/kontrollerinnimi/metodinnimi/@Html.Raw(item.Id)', {},
                    function (sdata) {
                        var tdata = new google.visualization.DataTable();
                        tdata.addColumn('date', 'Date');
                        tdata.addColumn('number', '@item.thing');
                        tdata.addColumn('number', '@item.target');

                        for (i = 0; i < sdata.length; ++i) {
                            tdata.addRows([[new Date(sdata[i][0]), parseFloat(sdata[i][1]), parseFloat(sdata[i][2]) ]]);
                        }
                        var @Html.Raw("options"+item.DivId) = {
                            series: {
                                0: { lineWidth: 2 },    // ensimmäisen linjan paksuus
                                1: { lineWidth: 1 }     // toisen linjan paksuus
                            },
                            colors:['@item.color', 'red'],
                            curveType: 'function',
                            legend: { position: 'bottom' },
                            vAxis: { minValue: 0, maxValue: 200, viewWindow: {min: 0, max: 200}}
                        };
                        var @Html.Raw("chart"+item.DivId) = new google.visualization.LineChart(document.getElementById('@Html.Raw("divchart"+item.DivId)'));
                        var temp = @Html.Raw("chart" + item.DivId);
                        temp.draw(tdata, @Html.Raw("options"+item.DivId));
                    });
                }
            </script>
            <a href="javascript:void" id="@item.Id" onclick="clickedMagn(this.id)"><div id="@Html.Raw("pvdivchart"+item.DivId)"></div></a>
            <a href="javascript:void" id="@item.Id" onclick="clicked(this.id)"><img src="~/Images/image.png" height="20" width="20" title="Muokkaa komponent
        </div>
    </div>
}
```

KUVA 16. Näkymän sisältämä JavaScript-koodi graafin piirtoa varten



KUVIO 1. JSON-tapahtuma näkymän ja tietokannan välillä

Kun komponentteja luotiin foreach-silmukoissa, muodostui aluksi ongelmaksi se, että graafi piirtyi oikein vain ensimmäiseen komponenttiin ja muut komponentit jäivät tyhjiksi. Ilmeni, että graafien piirtokomentojen tuli olla yksilöllisesti nimetyt, ja se ratkaistiin luomalla komponentin id:llä yksilöity väliaikaismuuttuja, kuten esimerkiksi kuvasta 16 ilmenee.

Graafeissa käytettiin viivadiagrammia, jonka viivojen paksuutta, tyyliä ja väriä voidaan muuttaa lähdekoodissa. Vaakasuora tavoitearvon viiva muodostetaan graafiin samalla tavoin kuin toinenkin, eikä sillä ole pakko olla lukuarvoa. Tavoitearvon viivan piirtoon haettiin ratkaisua myös graafiin muodostuvista tulkintaa helpottavista taustaviivoista (crosshair), mutta sitä ei työn aikana löydetty. Myös muun malliset diagrammit otettiin työssä puheeksi, mutta ne päätettiin jättää opinnäytetyön ulkopuolelle.

4.4 Testaus

Koska toteutusvaiheiden yksittäiset tavoitteet jätettiin alun suunnitteluvaiheessa vielä varsin avoimiksi ja eteneminen oli hyvin käytännönläheistä, ei varsinaisen testausdokumentin luomista nähty aiheelliseksi. Yksikkötestausta tehtiin jatkuvasti työn edetessä toiminnallisuuden varmistamiseksi.

Lähdekoodin toimivuutta testattiin työn aikana kolmella nykyisin käytetyimmällä selaimella (W3Schools Statistics 2015), jotka olivat Google Chrome (työssä käytetty versio 42.0.x), Mozilla Firefox (versio 37.0.x) ja Internet Explorer (versio 10). Toimivuutta muilla selaimilla ei ole testattu eikä taattu eikä työssä myöskään ole otettu huomioon skaalautuvuutta mobiililaitteille.

Käyttöliittymien käytön ja ulkoasun laatua arvioitiin heuristisesti koko kehitystyön ajan. Hyvänä muistilistana toimi 10-kohtainen lista niin kutsutuista Nielsenin säännöistä (Auer 2006), ja kaikki kohdat pyrittiin ottamaan mahdollisimman hyvin huomioon.

4.5 Dokumentointi

Työn alussa laadittiin projektisuunnitelma. Työn etenemistä dokumentoitiin itsenäisesti pitämällä Microsoft Excelissä erillistä työskentelypäiväkirjaa, johon kirjoitettiin päivittäin aikaansaannokset ja ilmenneet ongelmat. Opinnäytetyön loppuraportin kirjoitus aloitettiin hyvissä ajoin työn aikana asioiden ollessa vielä tuoreessa muistissa.

Toimeksiantajalla on portaalista oma järjestelmädokumenttinsa, johon päivitettiin graafinäkymän osuus. Toimeksiantaja ei tarvitse työstä muuta dokumentaatiota. Opinnäytetyön ulkopuolelle jätetyt asiat on otettu lähdekoodissa huomioon, niiden luomista helpottavia tekijöitä on jätetty koodiin kommentoituna tai ne on selvennetty erilliselle dokumentille.

5 POHDINTA JA YHTEENVETO

5.1 Työn odotukset, haasteet ja toteutumat

Työtä aloitettaessa oli jo hyvin selvillä, mitä lopputuloksen ulkonäöltä ja toiminnalta haluttiin, joten työvaiheiden suunnittelu ja toteutustapojen valikointi kävi helposti. Työskentely oli itsenäistä, mikä sopi työn luonteeseen hyvin. Eteneminen ja kehitystyö olivat pääosin ongelmattomia, koska Visual Studio oli kehitystyökaluna tuttu ja myös ASP.NET MVC:stä oli aiempaa kokemusta. Aikaa ei siis tarvittu käyttää ohjelmointiympäristön ja -kielten tai toimintatapojen opettelemiseen.

Pieniä haasteita ilmaantui työtä tehdessä, mutta niihin löydettiin varsin nopeasti soveltuvat ratkaisut. Graafinäkymän rakenteellinen toteutustekniikka täytyi työn puolivälissä vaihtaa toiseen, kun huomattiin, että tulevien työvaiheiden toteutus kävisi tarpeettoman monimutkaiseksi. Toteutustavan muuttaminen ei kuitenkaan aiheuttanut kovin suurta vaivannäköä tai aiemman työpanoksen mitätöitymistä, koska työssä oli osattu ottaa modulaarisuus ja yksinkertaisuus huomioon myös lähdekoodin rakenteessa.

JavaScript ja jQuery toivat työskentelyyn omat haasteensa, vaikka niistä olikin jo hieman aiempaa kokemusta. Kyseiset kehityskielet eivät itsessään kuulu oppilaitoksen opetussuunnitelmaan, mutta niiden itseopiskelu on kuitenkin helppoa, koska aineistoa ja ohjeita löytyy internetistä runsaasti.

Graafinäkymän kehityskohtia, kuten graafin tavoitearvon ottaminen mukaan, löydettiin työn edetessä. Kaikkia lopputuloksen toimivuuden kannalta tärkeitä kehityskohtia ei luonnollisestikaan osattu ottaa vielä työn alussa huomioon, mutta hyvin suunnitellun aikataulun vuoksi työn lopputulos ei kuitenkaan jäänyt puuttuvien toimintojen vuoksi vajaaksi. Loput kehityskohteista kuuluivat jatkokehitykseen ja ne osattiin jättää jäljellä olevan aikataulun ulkopuolelle.

Käyttöliittymän heuristisen arvioinnin muistilistana käytetyt Nielsenin säännöt (Auer 2006) täytettiin parhaan mukaan. Muistilistan kymmenestä kohdasta ainoaksi saavuttamattomaksi kohdaksi jäi tavoite käyttäjän muistikuorman vähentämisestä, jota työhön valittu autentikointitapa ei täytä kovin hyvin. Autentikoinnissa on kuitenkin otettu huomioon käyttöympäristön rajautuminen yrityksen sisäverkkoon, käyttäjän automaattisen uloskirjaamisajan siirtäminen myöhemmäksi ja ajatus portaalin kytkemisestä tulevaisuudessa AD-tunnistamisen piiriin.

Työn lopputuloksena on toimiva sovellus, joka vastaa omia ja toimeksiantajan odotuksia sekä työlle asetettuja vaatimuksia, jotka on esitetty kohdassa 4.1. Lähdekoodin dynaamisuudessa ja hyödynnettävyydessä saavutettiin hyväksyttävä taso ja eräät staattiset osuudet havaittiin dynaamisia vaihtoehtoja turvallisemmiksi.

5.2 Lopputuotteen käyttöönotto ja jatkokehitys

Toimeksiantaja aikoo ottaa lopputuotteen järjestelmätestaukseen opinnäytetyön päätyttyä. Onnistuneen testauksen päätteeksi lopputuote otetaan käyttöön ennaltamääräämättömänä ajankohtana. Kun tuote on todettu stabiiliksi paikallisessa käytössä, se aiotaan ottaa käyttöön pohjoismaisella tasolla yrityksen muissakin yksiköissä. Tuotetta tullaan myös kehittämään tarpeen mukaan.

Opinnäytetyön ulkopuolisia jatkokehitysideoita olivat muun muassa toisenlaisten diagrammityyppien valinta tarkasteltavan historiatiedon esittämiseen, mahdollisuus valita samaan graafiin useampia lähdetietoja vertailua varten, käyttäjätunnuksen salasanan resetointi sekä graafinäkymän ja asetusten käyttökielen vaihtaminen.

5.3 Aikataulu

Työ aloitettiin helmikuussa 2015 ja tavoitteena oli saada se päätökseen toukokuussa 2015. Työn alussa laaditun aikataulun loppuun jätettiin hieman joustovaraa, jos jokin työn projektisuunnitelmassa mainittu riski (liitteessä 1) toteutuisi. Aikataulussa pysymistä ja etenemistä seurattiin säännöllisin väliajoin pidetyissä etenemispalavereissa.

Työn määrä jaettiin kahden viikon mittaisiin työskentelyjaksoihin, joille annettiin suurpiirteiset tavoitteet. Tarkemmat työtavoitteet määrättiin päättyneen jakson toteutuneiden asioiden pohjalta, joten uusien tavoitteiden saavuttaminen ja etenemisen seuraaminen oli helppoa. Työn loppupuolella laadittiin lista jäljellä olevista työtavoitteista ja arvioitiin, mitkä niistä olisi vielä mahdollista saavuttaa ennen työn päättymistä.

Ammatilliset ja henkilökohtaiset taidot ja kyvyt otettiin aikataulua laadittaessa onnistuneesti huomioon. Aikataulussa pysyttiin suunnitellusti ja alkupään jaksojen työtavoitteet saavutettiin jopa etuajassa. Opinnäytetyö saatiin tavoitteen mukaisesti päätökseen.

5.4 Globaali hyöty

Työssä hyödynnettiin yleisiä ja hyväksi koettuja tekniikoita, jotka ovat jo tunnettuja ohjelmistokehityksen maailmassa. Käytetyistä tekniikoista löytyy runsaasti internetaineistoa ja kirjallisuutta.

Ohjelmistokehityksen maailmaan työ ei tarjoa kansainvälisessä mittakaavassa mitään uutta teknologiaa. Työn tarkotuksena olikin, että toimeksiantaja saa käyttöönsä sovelluksen, jota ei ole hankittu ulkopuoliselta, kaupalliselta toimittajalta.

5.5 Henkilökohtainen kehittyminen

Opinnäytetyö on ollut todella opettavainen ja mielenkiintoinen projekti, ja se on kartuttanut osaamista ohjelmistokehityksen saralta juuri oman mielenkiinnon ja henkilökohtaisen vahvuuden, käyttö-

liittymäsuunnittelun ja -toteutuksen parissa. Osaaminen myös teknisenä ohjelmoijana on vahvistunut, tiedonhakutaidot ovat kehittyneet ja tietämys ASP.NET:sta ja muista tässä työssä käytetyistä web-tekniikoista on syventynyt. On ollut hienoa huomata, että vastaan tulleisiin ohjelmointihaasteisiin osattiin etsiä totutusta poikkeaviakin ratkaisumalleja ja soveltaa aiemmin opittua tietoa entistä paremmin, mutta huomattiin myös se, että ratkaisu voi olla hyvinkin yksinkertainen ja se saattaa löytyä lähempää kuin on etsitty.

Opinnäytetyö on helpottanut muun muassa omien ammatillisten ja henkilökohtaisten kykyjen ja taitojen arviointia ja niiden ottamista huomioon esimerkiksi aikataulutuksissa. Työn aikana muistettiin myös, että ohjelmointiala on hyvin monipuolinen ja kehittyvä, ja suuntautumisvaihtoehtoja löytyy omien mieltymysten mukaan.

LÄHTEET JA TUOTETUT AINEISTOT

ASP.NET MVC FRAMEWORK RELEASE HISTORY 2015. Wikipedia. [Viitattu 2015-04-28.] Saatavissa: http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework#Release_history

ASP.NET MVC OVERVIEW 2015. [Viitattu 2015-04-27.] Saatavissa: https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx

AUER, Liisa 2006. Nielsenin säännöt. Virtuaaliammattikorkeakoulu. Opintojakso: Johdatus käytettävyyteen. [Viitattu 2015-05-04.] Saatavissa: http://www2.amk.fi/digma.fi/www.amk.fi/opintojaksot/030308/1111676348138/1111677021119/1161290796532/1161290917294.html

ECMA INTERNATIONAL 2006. C# Language Specification. [Viitattu 2015-04-28.] Saatavissa: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

ECMA INTERNATIONAL 2013. The JSON Data Interchange Format. [Viitattu 2015-04-27.] Saatavissa: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

ENFO OYJ 2010. ISO 9001 -laatusertifikaatti Service Deskille Suomessa. Tiedote [verkkojulkaisu]. [Viitattu 2015-02-24.] Saatavissa: http://www.enfo.fi/it-palvelut-ja-ulkoistukset/service-desk-ja-kayttajatukipalvelut/iso9001-laatusertifikaatti-service-deskille-suomessa/

ENFO OYJ 2011. Enfon Help Desk valittiin vuoden 2011 parhaaksi. Tiedote [verkkojulkaisu]. [Viitattu 2015-02-24.] Saatavissa: http://www.enfo.fi/pressrelease/enfon-help-desk-valittiin-vuoden-2011-parhaaksi/

ENFO OYJ 2012. Enfon Minna Nousiaisesta Suomen paras Service Desk -esimies 2012. Tiedote [verkkojulkaisu]. [Viitattu 2015-02-24.] Saatavissa: http://www.enfo.fi/pressrelease/enfon-minna-nousiaisesta-vuoden-service-desk-esimies-2012/

ENFO OYJ 2013. Vuoden paras asiakaspalvelija on Enfon Tero Hakkarainen. Tiedote [verkkojulkaisu]. [Viitattu 2015-02-24.] Saatavissa: http://www.enfo.fi/pressrelease/vuoden-paras-asiakaspalvelija-on-enfon-tero-hakkarainen/

ENFO OYJ 2014. Enfon Topias Laaksonen valittiin Vuoden Service Desk -asiantuntijaksi. Tiedote [verkkojulkaisu]. [Viitattu 2015-02-24.] Saatavissa: http://www.enfo.fi/pressrelease/enfon-topias-laaksonen-valittiin-vuoden-service-desk-asiantuntijaksi/

ENFOWAY 2014. The History of Enfo. [Viitattu 2015-04-28.] Saatavilla: http://www.enfoway.fi/fi/

ENTITY FRAMEWORK TUTORIAL 2015. What is Code-First? [Viitattu 2015-04-27.] Saatavissa: http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

GOOGLE LINE CHART 2015. Google Line Chart Data Policy. [Viitattu 2015-04-28.] Saatavissa: https://developers.google.com/chart/interactive/docs/gallery/linechart?hl=fi#Data_Policy

MSDN LAZY LOADING 2015. Entity Framework Loading Related Entities. [Viitattu 2015-04-27.] Saatavissa: https://msdn.microsoft.com/en-us/data/jj574232.aspx#lazy

RGB-COLOR VALUES 2011. [Viitattu 2015-04-14.] Saatavissa: http://www.w3.org/TR/css3-color/#rgb-color

RYYNÄNEN, Sauli 2015. Info- ja raporttiportaali. Savonia-ammattikorkeakoulu. Tietotekniikan koulutusohjelma. Opinnäytetyö. [Viitattu 2015-03-25.] Saatavissa: http://urn.fi/URN:NBN:fi:amk-201503213403

USING GOOGLE CHARTS 2015. [Viitattu 2015-03-25.] Saatavissa: https://developers.google.com/chart/interactive/docs/index

VISUAL STUDIO 2015. Visual Studio Express. [Viitattu 2015-05-04.] Saatavissa: https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx

VISUAL STUDIO APPLICATION DEVELOPMENT 2015. [Viitattu 2015-04-27.] Saatavissa: https://www.visualstudio.com/explore/application-development-vs

W3SCHOOLS STATISTICS 2015. Browser statistics since 2002. [Viitattu 2015-05-04.] Saatavissa: http://www.w3schools.com/browsers/browsers_stats.asp

W3SCHOOLS HTML 2015. Introduction to HTML. [Viitattu 2015-04-28.] Saatavissa: http://www.w3schools.com/html/html_intro.asp

W3SCHOOLS JS 2015. JavaScript Introduction. [Viitattu 2015-04-28.] Saatavissa: http://www.w3schools.com/js/js_intro.asp

W3SCHOOLS JQUERY 2015. jQuery Introduction. [Viitattu 2015-04-28.] Saatavissa: http://www.w3schools.com/jquery/jquery_intro.asp

VÄHÄKAINU, Kyösti 2015-02-24. Director, Service Desk. [Haastattelu.] Kuopio: Enfo Oyj.

LIITE 1: OPINNÄYTETYÖN RISKITEKIJÄT JA NIIDEN HALLINTA

Taulukossa 1 on listattu opinnäytetyöprojektin riskitekijöitä, ja pohdittu kuinka ne voitaisiin välttää. Riskitekijöiden vakavuusasteikko on 1-5, jossa 1 on ”hyvin vähäiset seuraukset” ja 5 on ”hyvin vakavat seuraukset”. Riskien toteutumisen todennäköisyysasteikko on 1-5, jossa 1 on ”hyvin epätodennäköinen” ja 5 on ”hyvin todennäköinen”. Vastuuhenkilö on opinnäytetyöntekijä.

TAULUKKO 1. Riskitekijät ja niiden hallinta.

Sairastumiset ja muut henkilökohtaiset tekijät					
Riskin ID	**Riskin kuvaus**	**Todennäköinen seuraus**	**Vakavuus-aste**	**Toden-näköisyys**	**Riskin hallinta**
1	Vastuuhenkilö sairastuu, lyhytaikainen sairasloma	Viivästyminen	4	3	Terveyden ylläpito ja runsas uni
2	Vastuuhenkilö sairastuu, pitkäaikainen sairausloma	Keskeytyminen	5	1	Ei vastuuhenkilön hallittavissa, mikäli sairausloman aiheuttaa vastuuhenkilöstä riippumaton tapaturmainen loukkaantuminen
3	Vastuuhenkilön motivaatio projektin eteenpäin viemiseen vähenee, tai ilmenee muita henkilökohtaisia hidasteita	Viivästyminen	3 tai 4	3	Pidetään mielessä, kuinka lähellä tutkintotodistuksen saaminen on
4	Vastuuhenkilön motivaatio projektin eteenpäin viemiseen loppuu kokonaan, tai ilmenee muita henkilökohtaisia esteitä	Keskeytyminen	5	2	Pidetään mielessä, kuinka lähellä tutkintotodistuksen saaminen on
5	Toimeksiantajan tai oppilaitoksen ohjaava henkilö sairastuu, lyhytaikainen sairausloma	Viivästyminen	3	3	Ei vastuuhenkilön hallittavissa
6	Toimeksiantajan tai oppilaitoksen ohjaava henkilö sairastuu, pitkäaikainen sairausloma	Viivästyminen	3 tai 4	1	Ei vastuuhenkilön hallittavissa Toimeksiantaja: Vastuuhenkilö jatkaa ilman ohjausta tai etsitään tilalle toinen ohjaava henkilö Oppilaitos: nimetään toinen ohjaava opettaja
7	Lopputyön kieliasun tarkastaja sairastuu	Viivästyminen	4	3	Ei vastuuhenkilön hallittavissa Oppilaitos nimeää toisen tarkastajan
8	Vastuuhenkilön osaaminen on puutteellista	Viivästyminen tai keskeytyminen	4	3	Tietoa etsitään runsaasti ja päivitetään teoreettista osaamista Pidetään toteutustavat tutuissa rajoissa ja suunnittelu realistisena, unohdetaan ne kaikkein villeimmät ideat jos ne vaativat täysin uutta tekniikkaa Projektin tehtäväkohdat pidetään selkeinä ja aikataulu joustavana Tarvittaessa muutetaan ratkaisutapaa ja pyydetään apua
Ulkoiset tekijät					
Riskin ID	**Riskin kuvaus**	**Todennäköinen seuraus**	**Vakavuus-aste**	**Toden-näköisyys**	**Riskin hallinta**
9	Projektitiedostot katoavat, varmuuskopio sisältää edellisen version	Lievä viivästyminen, mikäli edellisestä varmuuskopioinnista on kulunut aikaa	2	3	Säännölliset tiedostojen varmuuskopioinnit (verkkolevyt, ulkoinen massamuisti, sähköposti)
10	Projektitiedostot katoavat, varmuuskopioita ei ole tai nekin katoavat	Keskeytyminen, mikäli katoaminen tapahtuu projektin loppuvaiheessa	4 tai 5	1	Säännölliset tiedostojen varmuuskopioinnit (verkkolevyt, ulkoinen massamuisti, sähköposti)
11	Tietokone vaihdetaan, ja tarvittavat ohjelmistot joudutaan asentamaan uudelleen	Lievä viivästyminen	1 tai 2	2	Projektin tekoon vaadittavat ohjelmistot pidetään minimissä, jotta niiden asentamiseen kuluu mahdollisimman vähän aikaa
12	Projektitiedostot päätyvät vääriin käsiin	Lievä tietovuoto	3	2	Ulkoiset massamuistit pidetään tallessa,

					sähköpostien salasanoja ei luovuteta ulkopuolisille
13	Toimeksiantoyritys lopettaa toimintansa	Keskeytyminen	5	1	Ei vastuuhenkilön hallittavissa
14	Oppilaitos lopettaa toimintansa	Keskeytyminen	5	1	Ei vastuuhenkilön hallittavissa
15	Työn määrä on liian suuri suhteessa käytettävään aikaan	Suunniteltu työ jää toiminnoiltaan vajaaksi, pahemmassa tapauksessa viivästyminen tai keskeytyminen	4	4	Huolellinen aikataulutus ja työtehtävien modulointi Edistymisen jatkuva seuraaminen, muistetaan omat kyvyt suunnitteluvaiheessa (ks. ”Vastuuhenkilön osaaminen on puutteellista”)
16	Työn määrä on liian vähäinen suhteessa käytettävään aikaan	Vaikutus arvosanaan	2	1	Huolellinen aikataulutus ja edistymisen jatkuva seuraaminen Työtehtävien modulointi, laajennetaan osa-alueita tai kokeillaan jotain vaativampaa toteutustapaa